Universidad Carlos III de Madrid

Escuela Politécnica Superior (Leganés)

Grado en Ingeniería Informática (4er curso)

Asignatura: Sistemas de tiempo real

Práctica 1:

Programación de dispositivos y planificador cíclico simple

Elaborado por:

Responsable de software:
Gómez Aparicio, Alberto
(NIA: 363805 / Grupo 81 /
100363805@alumnos.uc3m.es)

Responsable de hardware:
Zhu, Zhenfeng
(NIA: 363798 / Grupo 81 /
100363798@alumnos.uc3m.es)

Responsable global:
Yánez Vargas, Jonathan
(NIA: 363885 / Grupo 81 /
100363885@alumnos.uc3m.es)

Leganés, Madrid. 7 de noviembre de 2019.

Índice de contenidos

Introducción

Este documento explica detalladamente el funcionamiento de las dos partes de la práctica y termina con una breve conclusión.

Parte software

Esta parte contendrá la explicación de la parte de software del sistema de control remoto que será el servidor implementado con VxWorks. El código de cada apartado individual es incremental, es decir, todo el código en el apartado A estará en el B con la consiguiente extensión para implementar su funcionalidad (del B), así pues, es C será mayor que el B y el último apartado, el D, contendrá todos los anteriores.

Apartado A

Para desarrollar la funcionalidad base, que consiste en mantener a la carretilla a una velocidad de 55km/h frenando si aumenta esa velocidad o acelerando si disminuye, son necesarias las siguientes tareas:

Tarea	Funcion	T=D	C	Identificador
Leer pendiente	task_slope()	10	0,9	A
Leer velocidad	task_speed()	10	0,9	B
ON/OFF acelerador	task_gas()	10	0,9	C
ON/OFF freno	task_brake()	10	0,9	D
ON/OFF mix	task_mix()	15	0,9	E

*Nota: la última columna, identificador, es el nombre que hemos dado a esa tarea para referirnos a ella (tanto en la memoria como en el código de la implementación).

Una vez tenemos las tareas y su información necesaria (periodo y tiempo de cómputo) pasamos a diseñar el planificador cíclico:

El uso de CPU es 0,42. Por lo tanto es factible. El ciclo principal es 30 segundos. Y aplicando el método general:

1. $\max(Ci) \leq Ts \Longrightarrow 0.9 \leq 10$
2. $Ts \geq \max(Ci) \ \forall i \Longrightarrow Ts \geq 0.9$
3. $\exists k \Longrightarrow Ti = k\ Ts \Longrightarrow Ti = [1, 2, 3, 5, 10]$
4. $Ts + (Ts - mcd(Ts, Ti)) \leq Ti$. Que se cumplen para todos los elementos de Ti

Por lo tanto la duración del ciclo secundario es de 5 segundos.

Entonces los ciclos secundarios, nombrados CSX, donde X es el número que indica cuál es (empezando en 1), tendrán las tareas:

CS1					CS2					CS3					CS4					CS5					CS6				
A	B	C	D		E					A	B				C	D				A	B	C	D		E				
3,6					0,9					1,8					1,8					3,6					0,9				

*Nota: la última fila son los segundos que tardan todas las tareas en el ciclo.

Una vez diseñado y planificado el problema, pasamos a su implementación. Explicaremos qué es lo que hace el proceso ligero que se encarga de todo, nombrado controlador:
A grandes rasgos es un bucle infinito que busca realizar las anteriores tareas separadas en ciclos secundarios (de aquí en adelante CS). Antes de este bucle inicializamos la variable que controla el tiempo del mezclador de la carretilla y la de control de tiempo del CS. El bucle tiene un switch para ejecutar el CS que toque en ese momento y al final le sumará uno (como hay 6 ciclos secundarios, la suma es circular para no sobrepasar nunca el 6); antes de terminar una vuelta del bucle, mediremos el tiempo que ha tardado el CS para dormir el tiempo sobrante del actual CS.

Las tareas son las siguientes:

- Leer pendiente: envía la cadena de texto que representa la petición del estado de la pendiente y actualiza una variable que guarda ese estado.
- Leer velocidad: similar a la anterior, pero actualiza la variable *speed* que es un entero.
- ON/OFF acelerador: utiliza la variable *speed* para decidir si hay que acelerar o dejar de acelerar con las condiciones del enunciado.
- ON/OFF freno: igual que la anterior pero con el objetivo de disminuir la velocidad.
- ON/OFF mix: al igual que las anteriores, activa o desactiva el mezclador con las restricciones de tiempo.

Apartado B

Para implementar el comportamiento de los focos, es necesario añadir dos tareas, una para leer la luz y otra para encender o apagar los focos:

Tarea	Función	T=D	C	Identificador
Leer pendiente	task_slope()	10	0,9	A
Leer velocidad	task_speed()	10	0,9	B
ON/OFF acelerador	task_gas()	10	0,9	C
ON/OFF freno	task_brake()	10	0,9	D
ON/OFF mix	task_mix()	15	0,9	E
Leer luz	task_brightness()	5	0,9	F
ON/OFF focos	task_lights()	5	0,9	G

Hemos reducido el periodo de estas nuevas tareas (G y H) de 6 segundos a 5 para

que sean múltiplo de las demás y sea más sencillo de planificar.
Con un uso de CPU de 78%, el ciclo principal de 30 segundos y los secundarios con el método general:

1. $\max(Ci) \leq Ts \Longrightarrow 0.9 \leq 5$
2. $Ts \geq \max(Ci)\ \forall i \Longrightarrow Ts \geq 0.9$
3. $\exists k \Longrightarrow Ti = k\ Ts \Longrightarrow Ti = [1, 2, 3, 5]$
4. $Ts + (Ts - mcd(Ts, Ti)) \leq Ti$. Que se cumplen para todos los elementos de Ti

Por lo tanto la duración del ciclo secundario es de 5 segundos.

CS1					CS2					CS3					CS4					CS5					CS6				
A	B	F	G	E	C	D	F	G		A	B	F	G		C	D	F	G	E	A	B	F	G		C	D	F	G	
4,5					3,6					3,6					4,5					3,6					3,6				

El código de estas tareas es sencillo; para leer la luz solo hay que hacer una petición del valor de brillo de luz y para encender los focos: utilizando el valor de la respuesta, activar o desactivar las luces con la condición de que el brillo supere o no un 50%.

Apartado C

Ahora hay que dotar al servidor de tres modos diferentes de comportamiento para la carretilla:

- Modo normal: que es el comportamiento de los anteriores apartados.
- Modo de frenada: cuando la distancia al depósito sea menor que el valor prefijado, del modo normal se pasa a este.
- Modo de descarga: si la distancia al punto es cero y la carretilla va lo suficientemente despacio, frenará y pasaremos al modo de descarga. Se puede volver al modo normal cuando el sensor indique el fin.

Modo Normal				
Tarea	**Función**	**T=D**	**C**	**Identificador**
Leer pendiente	task_slope()	10	0,9	A
Leer velocidad	task_speed()	10	0,9	B
Leer distancia	task_distance()	10	0,9	C
ON/OFF acelerador	task_gas()	10	0,9	D
ON/OFF freno	task_brake()	10	0,9	E
ON/OFF mix	task_mix()	15	0,9	F
Leer luz	task_brightness()	5	0,9	G
ON/OFF focos	task_lights()	5	0,9	H

*Nota: al igual que en apartado B, hemos reducido el periodo de las tareas G y H.

El uso de CPU es del 87%, con un ciclo principal de 30 s y 5 s de ciclo secundario:

1. $\max(Ci) \leq Ts \Longrightarrow 0.9 \leq 5$
2. $Ts \geq \max(Ci) \ \forall i \Longrightarrow Ts \geq 0.9$
3. $\exists k \Longrightarrow Ti = k\ Ts \Longrightarrow Ti = [1, 2, 3, 5]$
4. $Ts + (Ts - mcd(Ts, Ti)) \leq Ti$. Que se cumplen para todos los elementos de Ti

Por lo tanto la duración del ciclo secundario es de 5 segundos.

CS1					CS2					CS3					CS4					CS5					CS6				
A	B	C	G	H	D	E	F	G	H	A	B	C	G	H	D	E	F	G	H	A	B	C	G	H	D	E	G	H	
4,5					4,5					4,5					4,5					4,5					3,6				

Modo Frenada				
Tarea	**Función**	**T=D**	**C**	**Identificador**
Leer pendiente	task_slope()	10	0,9	A
Leer velocidad	task_speed()	5	0,9	B
Leer distancia	task_distance()	10	0,9	C
ON/OFF acelerador	task_gas()	5	0,9	D
ON/OFF freno	task_brake()	5	0,9	E
ON/OFF mix	task_mix()	15	0,9	F
ON focos	task_on_lights()	30	0,9	G

El uso de CPU es del 81%, con un ciclo principal de 30 s y de ciclo secundario:

1. $\max(Ci) \leq Ts \Longrightarrow 0.9 \leq 5$
2. $Ts \geq \max(Ci) \ \forall i \Longrightarrow Ts \geq 0.9$
3. $\exists k \Longrightarrow Ti = k\ Ts \Longrightarrow Ti = [1, 2, 3, 5]$
4. $Ts + (Ts - mcd(Ts, Ti)) \leq Ti$. Que se cumplen para todos los elementos de Ti

El ciclo secundario dura 5 segundos.

CS1					CS2					CS3					CS4					CS5					CS6				
B	D	E	A	C	B	D	E	F		B	D	E	A	C	B	D	E	F		B	D	E	A	C	B	D	E	G	
4,5					3,6					4,5					3,6					4,5					3,6				

Modo Descarga				
Tarea	**Función**	**T=D**	**C**	**Identificador**
Leer fin descarga	task_finish_unload()	5	0,9	A
ON/OFF mix	task_mix()	15	0,9	B
ON focos	task_on_lights()	5	0,9	C

El uso de CPU es del 42%, con un ciclo principal de 15 segundos y de ciclo secundario:

1. $\max(Ci) \leq Ts \Longrightarrow 0.9 \leq 5$
2. $Ts \geq \max(Ci)\ \forall i \Longrightarrow Ts \geq 0.9$
3. $\exists k \Longrightarrow Ti = k\ Ts \Longrightarrow Ti = [1, 2, 3, 5]$
4. $Ts + (Ts - mcd(Ts, Ti)) \leq Ti$. Que se cumplen para todos los elementos de Ti

El ciclo secundario dura 5 segundos también.

CS1					CS2					CS3				
A	C				A	C	B			A	C			
1,8					2,7					1,8				

Hay pequeñas modificaciones a las tareas del apartado B, que es el que se toma como base para hacer este:

- En el bucle del controlador ahora hay que comprobar el modo en el que estamos para ejecutar las tareas correspondientes.
- En la tarea de *ON/OFF acelerador* hay que incluir una condición para que si estamos en el modo de frenada, la velocidad a mantener sea 2,5 m/s; no a 55 m/s como en el modo normal.
- En *ON/OFF freno*, similar a la anterior modificación, pero en vez de dar/soltar gas, activar/desactivar el freno.

Las nuevas tareas que hacen falta son:

- *Leer distancia*: hacemos una petición para obtener el valor de la distancia al siguiente depósito y cambiamos el modo si:
  - Si estamos en modo normal y la distancia es menor a la debida ⟶ Pasamos al modo frenada.
  - Si estamos en modo frenada, la distancia es cero y vamos a menor velocidad que 10 m/s ⟶ Pasamos al modo de descarga.

  Si hemos cambiado de modo, devolvemos el entero con valor 1 para que el bucle del controlador comience a ejecutar las tareas del nuevo modo del primer ciclo secundario.
- *ON focos*: simplemente enciende los focos (sin tener el cuenta el valor de luz).
- *Leer fin descarga*: el sensor que indica el fin de descarga nos responderá con la cadena oportuna cuando debamos abandonar el modo y pasar al normal.

Apartado D

En este último apartado, comprobaremos si las tareas se han podido ejecutar dentro del tiempo máximo permitido, si no lo hacen, pasaremos al modo de emergencia.

Tarea	Funcion	T=D	C	Identificador
Leer pendiente	task_slope()	10	0,9	A
Leer velocidad	task_speed()	10	0,9	B
OFF acelerador	task_off_gas()	10	0,9	C
ON freno	task_on_brake()	10	0,9	D
ON/OFF mix	task_mix()	15	0,9	E
ON focos	task_on_lights()	10	0,9	F
Activar modo Arduino (1 vez)	task_emergence_mode()	10	0,9	G

Este apartado tiene un uso de CPU del 60%, con el ciclo principal de 30 segundos y de los secundarios:

1. $\max(Ci) \leq Ts \Longrightarrow 0.9 \leq 10$
2. $Ts \geq \max(Ci) \ \forall i \Longrightarrow Ts \geq 0.9$
3. $\exists k \Longrightarrow Ti = k\ Ts \Longrightarrow Ti = [1, 2, 3, 5, 10]$
4. $Ts + (Ts - mcd(Ts, Ti)) \leq Ti$. Que se cumplen para todos los elementos de Ti

En el modo de emergencia cada ciclo secundario dura 10 segundos.

CS1										CS2										CS3									
A	B	C	D	F	G		E			A	B	C	D	F	G					A	B	C	D	F	G		E		
6,3										5,4										6,3									

Para comprobar si una tarea ha tardado más de lo previsto simplemente mediremos cuanto a demorado, y si pasa, activaremos el modo con las nuevas tareas:

- *OFF acelerador*: que desactiva el gas.
- *ON freno*: que activa el freno.
- *Activar modo Arduino (1 vez)*: que envía la cadena de texto correspondiente a la petición de inicio de emergencia.

Parte hardware

Para obtener el tiempo de cómputo de cada tarea, utilizamos la función *micros()* para medir el tiempo que ha durado cada tarea en su ejecución, cogiendo el peor de los casos.

En cuanto a la implementación, se declaran variables globales para representar el estado de cada tarea del sistema. Se comprueba sus estados de forma periódico en el bucle del programa arduino y se cambia sus valores de acuerdo a los mensajes recibidos u otras condiciones.

Apartado A

Todas las tareas tienen tiempos de cómputo muy bajos, por lo que hemos decidido representarlos en la tabla por microsegundos.

El periódo principal es el plazo de respuesta de la tarea A, porque tiene mayor valor. Como son periodos armónicos, dentro del periodo principal nos encontramos con dos periodos secundarios a valor de 10.000.

Como los tiempos de cómputo de las tareas son relativamente pequeños que el plazo de respuesta, en un periodo secundario puede caber todas las tareas.

De este modo, las tareas B - F deben repetir en cada periodo secundario, y la tarea A debe repetir una sola vez en el periodo principal.

Índice	Tarea (Enunciado)	Tarea (Código)	T (us)	C (us)
A	Servidor de Comunicaciones	comm_server()	200000	276
B	Act/Desact Acelerador	check_accel()	100000	8
C	Act/Desact Freno	check_brk()	100000	8
D	Act/Desact Mixer	check_mix()	100000	8
E	Calcular y mostrar Velocidad	check_slp()	100000	16
F	Leer Pendiente	represent_speed()	100000	96
		Utilización del CPU	0,00274	
		Ciclo principal (CP)	200000	
		Ciclo secundario (CS)	100000	

CP1																				
CS1											CS2									
A	B	C	D	E	F						B	C	D	E	F					
0									100000											200000

Apartado B

Al igual que el apartado A, las tareas G y H también tienen tiempos de cómputo muy pequeños, de forma que se puede agrupar todas las tareas en un mismo periodo secundario.

Entonces hemos aplicado el mismo planificador de A añadiendo las dos tareas adicionales para cada periodo secundario.

Índice	Tarea (Enunciado)	Tarea (Código)	T (us)	C (us)
A	Servidor de Comunicaciones	comm_server()	200000	276
B	Act/Desact Acelerador	check_accel()	100000	8
C	Act/Desact Freno	check_brk()	100000	8
D	Act/Desact Mixer	check_mix()	100000	8
E	Calcular y mostrar Velocidad	check_slp()	100000	16
F	Leer Pendiente	represent_speed()	100000	96
G	Leer sensor de luminosidad	check_lit()	100000	8
H	Act/Desact Focos	check_lam()	100000	160
		Utilización del CPU	0,00442	
		Ciclo principal (CP)	200000	
		Ciclo secundario (CS)	100000	

CP1																				
CS1											CS2									
A	B	C	D	E	F	G	H				B	C	D	E	F	G	H			
0									100000											200000

Apartado C

En este apartado dispone de tres modos de ejecución, cada modo dispone de sus propias tareas a ejecutar, de las cuales las tareas en color gris son las que se repiten en todos los modos, mientras las tareas en color negra son las propias del modo.

Al igual que el apartado A y B, las tareas adicionales a añadir pueden caber perfectamente en un mismo ciclo secundario, por tanto hemos aplicado el mismo planificador que el apartado A y B.

Modo de selección de distancia				
Índice	**Tarea (Enunciado)**	**Tarea (Código)**	**T (us)**	**C (us)**
A	Servidor de Comunicaciones	comm_server()	200000	276
B	Act/Desact Acelerador	check_accel()	100000	8
C	Act/Desact Freno	check_brk()	100000	8
D	Act/Desact Mixer	check_mix()	100000	8
E	Calcular y mostrar Velocidad	check_slp()	100000	16
F	Leer Pendiente	represent_speed()	100000	96
G	Leer sensor de luminosidad	check_lit()	100000	8
H	Act/Desact Focos	check_lam()	100000	160
I	Selector de la distancia	check_deposit_distance()	100000	164
J	Mostrar la distancia seleccionada	display_distance_segment()	100000	4
K	Validad distancia seleccionada	validation_distance()	100000	8
		Utilización del CPU	0,00618	
		Ciclo principal (CP)	200000	
		Ciclo secundario (CS)	100000	

CP1																									
CS1														CS2											
A	B	C	D	E	F	G	H	I	J	K				B	C	D	E	F	G	H	I	J	K		
0												100000													200000

En cuanto a la implementación, además de incluir todas las tareas de los apartados anteriores (en grises), se encarga también de leer la distancia del potenciómetro en un rango proporcional de 10000 a 90000 y mostrar la primera cifra del número en el display de 7 segmentos.

También comprueba si se ha pulsado el botón, si es el caso guarda el valor de distancia que tiene el selector en ese momento como distancia que queda por llegar al depósito, en una variable, y cambia al modo de acercamiento al depósito.

Modo de acercamiento al depósito				
Índice	**Tarea (Enunciado)**	**Tarea (Código)**	**T (us)**	**C (us)**
A	Servidor de Comunicaciones	comm_server()	200000	276
B	Act/Desact Acelerador	check_accel()	100000	8
C	Act/Desact Freno	check_brk()	100000	8
D	Act/Desact Mixer	check_mix()	100000	8
E	Calcular y mostrar Velocidad	check_slp()	100000	16
F	Leer Pendiente	represent_speed()	100000	96
G	Leer sensor de luminosidad	check_lit()	100000	8
H	Act/Desact Focos	check_lam()	100000	160
J	Mostrar la distancia seleccionada	display_real_distance_deposit()	100000	108
		Utilización del CPU	0,0055	
		Ciclo principal (CP)	200000	
		Ciclo secundario (CS)	100000	

CP1																									
CS1													CS2												
A	B	C	D	E	F	G	H	J					B	C	D	E	F	G	H	J					
0												100000													200000

En este modo, además de ejecutar las tareas de A y B (en grises), calcula periódicamente la distancia que queda por llegar al depósito, e imprime la distancia en el display de 7 segmentos.

Modo de parada				
Índice	**Tarea (Enunciado)**	**Tarea (Código)**	**T (us)**	**C (us)**
A	Servidor de Comunicaciones	comm_server()	200000	276
B	Act/Desact Acelerador	check_accel()	100000	8
C	Act/Desact Freno	check_brk()	100000	8
D	Act/Desact Mixer	check_mix()	100000	8
E	Calcular y mostrar Velocidad	check_slp()	100000	16
F	Leer Pendiente	represent_speed()	100000	96
G	Leer sensor de luminosidad	check_lit()	100000	8
H	Act/Desact Focos	check_lam()	100000	160
L	Leer fin de parada	read_end_stop()	100000	12
		Utilización del CPU	0,00454	
		Ciclo principal (CP)	200000	
		Ciclo secundario (CS)	100000	

CP1																									
CS1													CS2												
A	B	C	D	E	F	G	H	L					B	C	D	E	F	G	H	L					
0												100000													200000

En este modo se encarga de comprobar si se ha pulsado el botón para salir del modo de parada y pasar al modo normal. La velocidad es siempre 0 en este modo.

Apartado D

Este apartado es idéntico al apartado C pero añade un modo de emergencia que excluye la tarea de lectura del sensor de luminosidad, pues mantiene los focos activados durante todo el tiempo.

Modo de emergencia				
Índice	**Tarea (Enunciado)**	**Tarea (Código)**	**T (us)**	**C (us)**
A	Servidor de Comunicaciones	comm_server()	200000	276
B	Act/Desact Acelerador	check_accel()	100000	8
C	Act/Desact Freno	check_brk()	100000	8
D	Act/Desact Mixer	check_mix()	100000	8
E	Calcular y mostrar Velocidad	check_slp()	100000	16
F	Leer Pendiente	represent_speed()	100000	96
H	Act/Desact Focos	check_lam()	100000	160
		Utilización del CPU	0,00178	
		Ciclo principal (CP)	200000	
		Ciclo secundario (CS)	100000	

CP1																									
CS1													CS2												
A	B	C	D	E	F	H							B	C	D	E	F	H							
0												100000													200000

En cuanto a la implementación, se encarga, además de las tareas de los apartados anteriores, a activar el freno y los focos durante todo el tiempo, y mantener desactivado el acelerador.

Conclusiones y comentarios personales

Hemos aprendido mucho acerca de programar en Arduino, lo cual nos ha abierto una nueva visión al mundo de electrónica.

Por otro lado, hemos conseguido entender mejor las planificaciones cíclicas, por aplicar los conocimientos aprendidos de la teoría en la vida real.

Tenemos que agradecer mucho a los profesores, quienes nos resuelven las dudas constantemente. Especialmente al profesor Javier Fernández muñoz, quien estuvo muy paciente con nosotros para las dudas que le planteamos.